API Documentation
PGA 9500
OPC UA

Rego-Fix AG, Version: 1.1.4

This API (Application Programming Interface) documentation offers comprehensive guidance for the integration of the PGA 9500 into automation cells It allows you to control various actions and retrieve important statistics, errors, maintenance information and notifications from the PGA 9500. Whether you are developing software, building automation systems, or troubleshooting the PGA unit, this documentation will assist you in understanding the API commands and their corresponding functionalities.

Get ready to explore the capabilities of the PGA communication interface and unlock the full efficiency potential of the PGA 9500.

Index

1. Operating modes

1.1. Automatic (default)

This mode allows all the necessary commands to be executed via the PGA interface. The automatic mode should be set by default, otherwise it would have to be set via the HMI interface to use the PGA in an automated system with all its functions and be able to execute them via the main controller with OPC UA protocol. In the event of a fault, the unit will stop and report the error via status code. The PGA is automatically initialized after enabling (setEnable) and is ready for use.

1.2. Manual

This mode can be set via the HMI interface. Once set, this mode must be confirmed with 'reset'. In this mode, all functions can be performed manually and separately via the HMI interface. For example, to check individual processes in the event of an error. Individual settings can also be made this way. If the PGA is not initialized, this process must be started manually.

2. Tool-change handling

To supply the PGA 9500 with a cutting tool/collet/toolholder, an additional handling system is required, which can be chosen by the system integrator. This could be a robot or an alternative feeding system. The controller of the handling system can communicate with the PGA 9500 via a network connection in the OPC UA architecture and control the commands of the PGA interface.

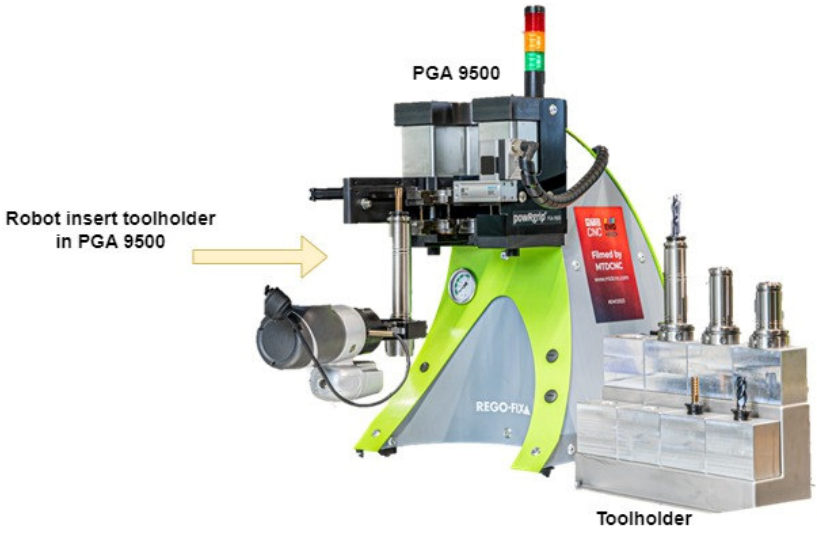

Fig.1: Visualization of the PGA tool-change handling

3. Connection setup

To establish a seamless communication link between the PGA 9500 and external systems, the connection setup of the PGA 9500 plays a crucial role. To establish a connection with the PGA 9500, you need to follow a series of steps.

- First, ensure that the unit is powered on and connected to the network. The PGA 9500 can be integrated into various automation setups as it supports Ethernet communication.
- Once the unit is ready, you can proceed to configure the connection parameters. These settings determine how external systems can interact with the unit. The protocol used is OPC UA (Open Platform Communications Unified Architecture), which provides a

standardized and secure method of communication. To initiate the connection, external systems must make a connection request using the configured IP address and port number.

- Once connected, bi-directional communication can take place. During the connection, external systems can send commands to the PGA 9500 to control its operations. The PGA 9500 processes the commands and sends back responses, providing real-time information and notifications. It's important to handle connection errors and timeouts gracefully. The documentation provides details on error handling and troubleshooting tips to ensure a robust and stable connection. By following the connection setup guidelines in this documentation, you'll be able to seamlessly integrate the PGA 9500 into your automation environment for efficient control and monitoring.

When you receive your new PGA 9500, the IP address of the device is: http://192.168.1.69. To check if the connection can be established, it is possible to ping the device. If the ping is successful, it should be possible to connect to the device's user interface using a web browser (Google Chrome, Firefox, Edge...) with the address http://192.168.1.69:8080/webvisu.htm.

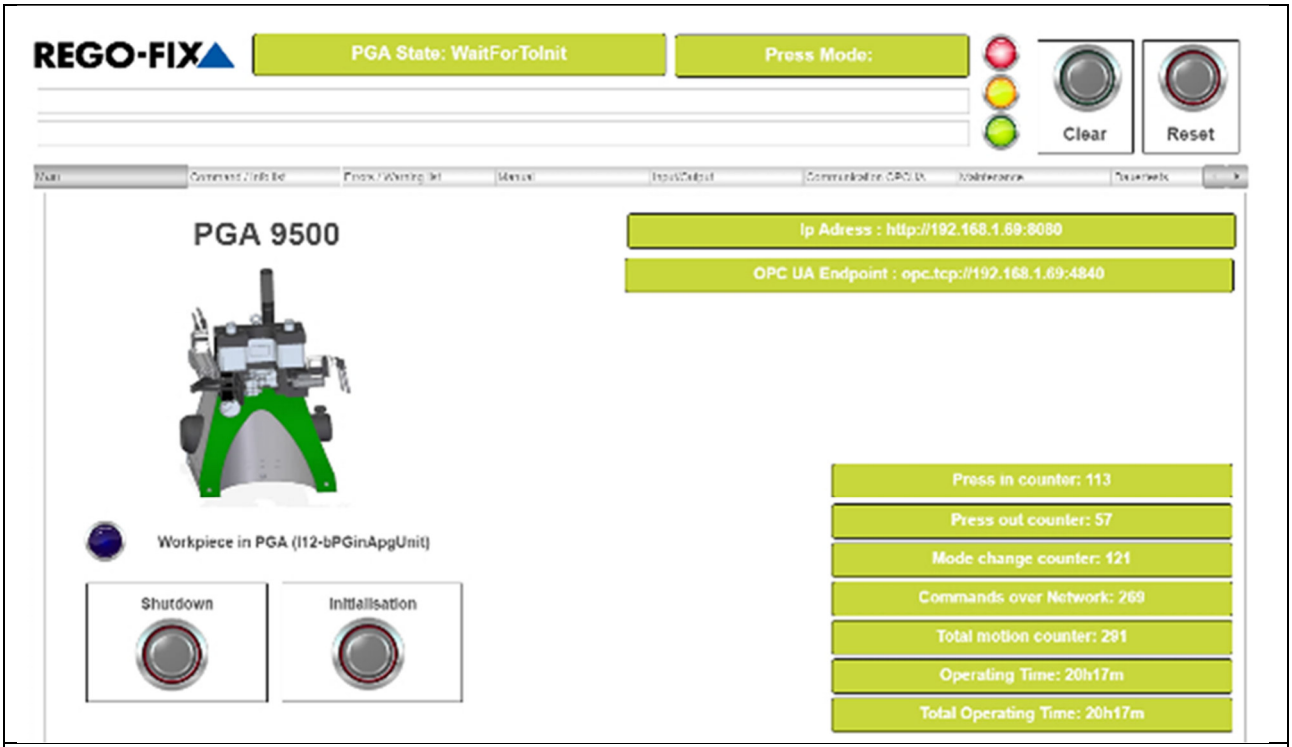

Fig.2: Start/Home Page PGA Visualization (HMI). If this page is displayed, the connection to the PGA 9500 has been established successfully.

3.1. Check the connection to the OPC UA Server with OPC UA Client

To check if an OPC UA communication can be established, an external client can be used named UA Expert. It is available on https://www.unified-automation.com/downloads/opc-ua-clients.html

3.2. OPC UA connection

Endpoint IP: opc.tcp://192.168.1.69
Endpoint: opc.tcp://192.168.1.69:4840

Endpoint over name:
Endpoint: 'opc.tcp://RevPi77889:4840'
Endpoint: 'opc.tcp://RevPiXXXXX:4840'

'x' is a placeholder for the serial number of the integrated PLC. This number can be found on the front of the control cabinet.

Fig.3: Connection address on the PGA control cabinet

The QR-code on the sticker of the control cabinet will guide you to the Github website (www.github.com).

3.3. UaExpert

UAExpert Client is a powerful software tool used for testing and troubleshooting OPC UA servers and clients. With UAExpert Client, users can easily connect to OPC UA servers, browse their address space, read and write data, and monitor server performance. The tool supports various security mechanisms, including user authentication and encryption, ensuring secure communication between the client and server. With its intuitive user interface and comprehensive set of features, UAExpert Client empowers users to efficiently validate OPC UA implementations, diagnose issues, and optimize the performance of their OPC UA systems.

3.4. UaExpert server configuration

To connect to a OPC UA server, the user needs to press the “plus” sign and add a new server to the project.

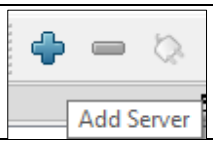

Fig.4: Add server

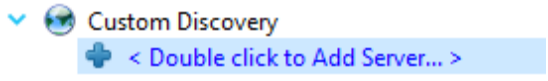

Fig.5: Custom Discovery

In the server selection window, you need to double-click on the add server layer. There are several subsections. It is important to choose the customer discovery section.

A window appears on the screen. There you need to enter the endpoint IP address which is given by initialization, or you choose it by your own. The OPC UA connection can be found in the section “OPC UA connection”.

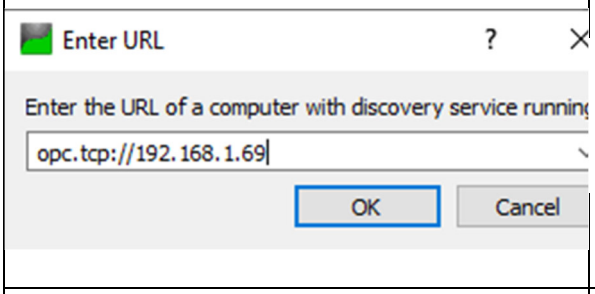

Fig.6: Endpoint IP address

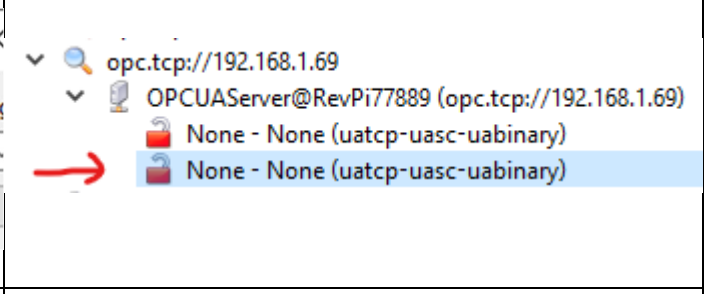

Fig.7: Server with name

When there is an OPC UA server found, a server icon with the server's name and IP address appears below the IP address. If this doesn't pop up there could be a problem with the connection to the device. Check the network adaptor configurations and have a look in the troubleshooting section. Press OK to continue.

In the project there is now added a connection to the OPC server. Normally the connection establishes by itself. If it does not connect by itself press the connection button.

Fig.8: Connection in the project

When everything works fine the connection is established and you can see the same structure as in *Fig.9* below. You can expand the folder root like below to the subfolder "PGA_Interface". In this folder are all variables to control the PGA device over a network connection.

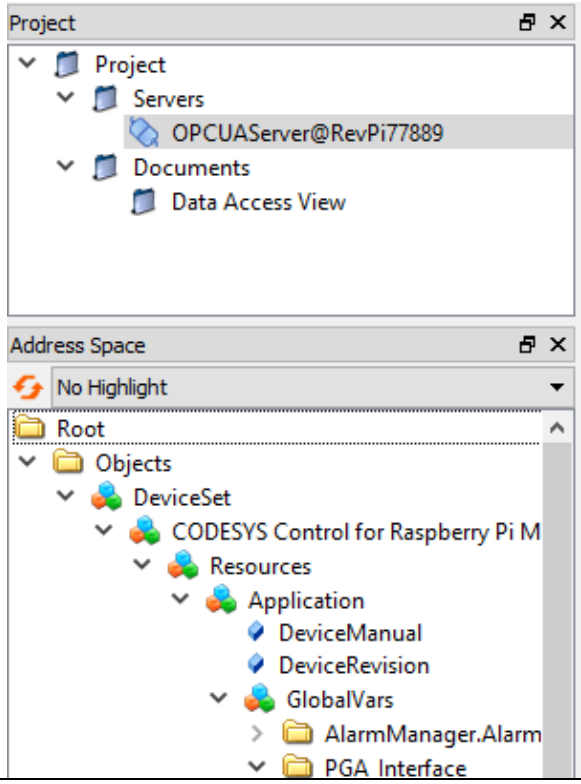

Fig.9: Project section

3.5. UaExpert hierarchy

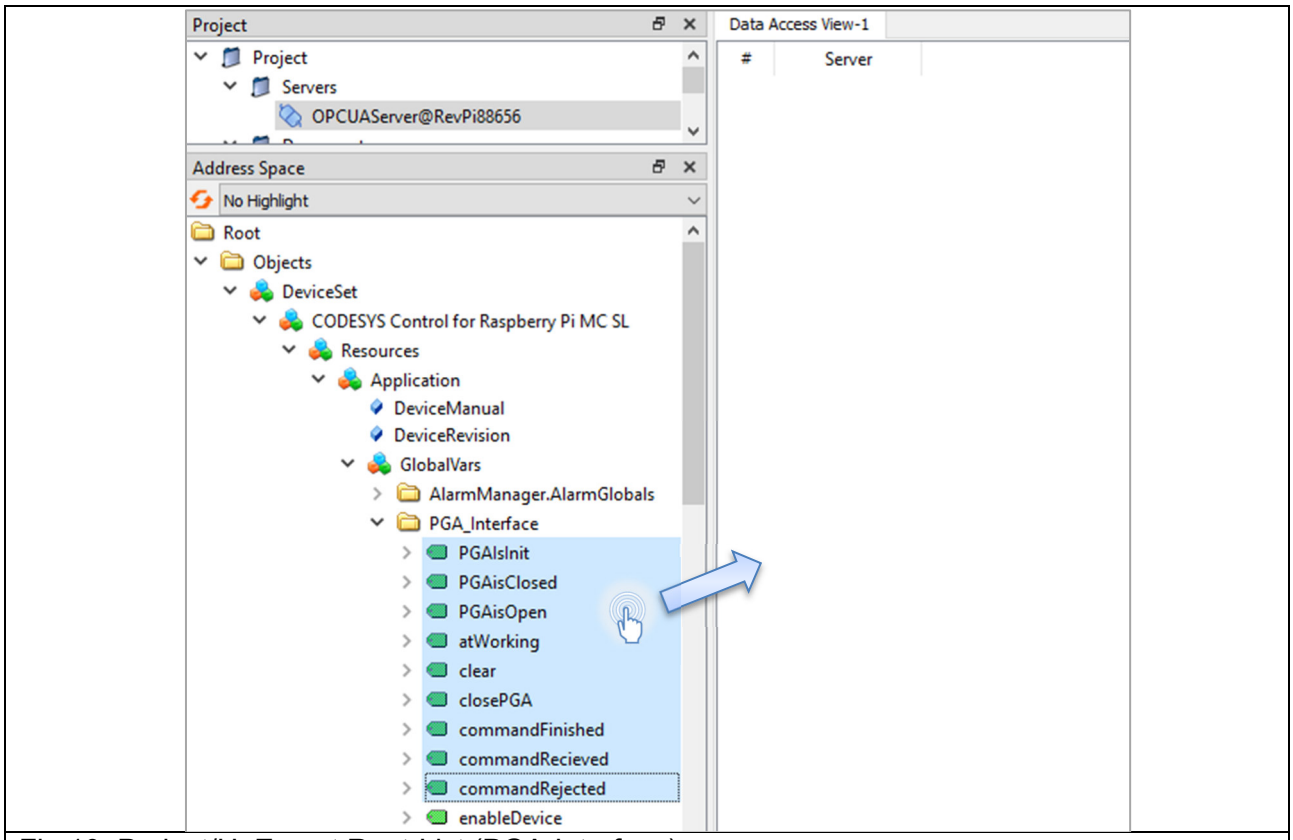

Fig.10: Project/UaExpert Root-List (PGA-Interface)

To visualize the data you can drag and drop the variables to the middle section of the window.

3.6. UaExpert example data

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	PGAIsInit	false	Boolean	14:15:41.547	14:15:41.547	Good
2	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	PGAisClosed	false	Boolean	14:15:41.547	14:15:41.547	Good
3	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	PGAisOpen	false	Boolean	14:15:41.547	14:15:41.547	Good
4	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	atWorking	true	Boolean	14:15:41.547	14:15:41.547	Good
5	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	clear	false	Boolean	14:15:41.547	14:15:41.547	Good
6	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	closePGA	false	Boolean	14:15:41.547	14:15:41.547	Good
7	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	commandFinis...	false	Boolean	14:15:41.547	14:15:41.547	Good
8	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	commandRecie...	false	Boolean	14:15:41.547	14:15:41.547	Good
9	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	commandRejec...	false	Boolean	14:15:41.547	14:15:41.547	Good
10	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	enableDevice	false	Boolean	14:15:41.547	14:15:41.547	Good
11	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	error	0	UInt16	14:15:41.547	14:15:41.547	Good
12	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	execute	false	Boolean	14:15:41.547	14:15:41.547	Good
13	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	isEnabled	false	Boolean	14:15:41.547	14:15:41.547	Good
14	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	maintenance	false	Boolean	14:15:47.074	14:15:47.074	Good
15	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	manual	false	Boolean	14:15:47.074	14:15:47.074	Good
16	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChange	false	Boolean	14:15:47.074	14:15:47.074	Good
17	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChangeToIn	false	Boolean	14:15:47.074	14:15:47.074	Good
18	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChangeTo...	false	Boolean	14:15:47.074	14:15:47.074	Good
19	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	modeChangeW...	false	Boolean	14:15:47.074	14:15:47.074	Good
20	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	notifications	0	UInt16	14:15:47.074	14:15:47.074	Good
21	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	openPGA	false	Boolean	14:15:47.074	14:15:47.074	Good
22	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	pause	false	Boolean	14:15:47.074	14:15:47.074	Good
23	OPCUAServer@RevPi...	NS4|String||var|CODESYS ...	paused	false	Boolean	14:15:47.074	14:15:47.074	Good

Fig.11: Example - Data Access View in UA-Expert, PGA-Interface Variables

Over the data access view in UA Expert the variables can be manipulated and tested out. In the “status-code-raw” column the connection state can be seen. Variables can be set by clicking. Integers have numbers to change, while Booleans have a checkbox to set or unset.

4. Data model

The Data Model is a foundational concept in database management that defines how data is organized, structured and related within a system. It serves as a blueprint for data storage and manipulation, enabling efficient data retrieval and manipulation operations.

4.1. Communication API

1: PGA State
2: PGA Response
3: Interaction Commands
4: Command 1
5: Command 2

4.1.1. Permissions:

id	Desciption
r	read
w	write
rw	read + write

4.1.2. Variable list

Variable name	Datatype	Permissions	OPC UA Connection String
isEnable	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.isEnabled*
maintenance	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.maintenance*
Error	Uint	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.error*
safety	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.safety*
notifications	Uint	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.notifications*
pressInMode	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pressInMode*
pressOutMode	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pressOutMode*
readyForCommands	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.readyForCommands*
commandRecieved	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.commandRecieved*
atWorking	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.atWorking*

Variable name	Datatype	Permis-sions	OPC UA Connection String
PGAisClosed	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.PGAisClosed*
PGAisOpen	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.PGAisOpen*
PGinAPGunit	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.PGinAPGunit*
paused	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.paused*
commandRejected	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.commandRejected*
commandFinished	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.commandFinished*
pressFinish	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pressFinish*
PGAIsInit	Bool	r	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.PGAIsInit*
enableDevice	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.enableDevice*
spaceFree	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.spaceFree*
execute	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.execute*
reset	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.reset*
pause	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pause*
Clear	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.clear*
shutdown	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.shutdown*
pressInComplete	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pressInComplete*
pressOutComplete	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pressOutComplete*
pressInIncomplete	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pressInIncomplete*
pressOutIncomplete	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.pressOutIncomplete*
closePGA	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.closePGA*

Variable name	Datatype	Permis-sions	OPC UA Connection String
openPGA	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.openPGA*
modeChange	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.modeChange*
modeChangeWith-Door	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.modeChangeWithDoor*
modeChangeToIn	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.modeChangeToIn*
modeChangeToOut	Bool	w	*NS4|String||var|CODESYS Control for Raspberry Pi MC SL.Application.PGA_Interface.modeChangeToOut*

4.1.3. Description of the variables

- **Read**

Is Enabled
This Bool is used to check if the PGA is enabled. If the PGA is enabled, the PGA will respond with True. If the PGA is not enabled, the PGA responds with False.

Maintenance
This Bool is used to check if the PGA is in the maintenance mode or not. The PGA will respond with True if the PGA is in Maintenance. If the PGA is not in Maintenance, the PGA responds with False.

Error
This unsigned integer is used to check if there is an error in the PGA and to indicate this with of a code. If the PGA has an error it will return a value >0, if there is no error the value remains 0.

Safety
This Bool is used to check whether the PGA safety system is closed or open. If the PGA is in a safe state, the PGA will respond with True. If the PGA is not in a safe state, the PGA will respond with a False.

Notifications
This unsigned integer is used to check if the PGA has a notification and to indicate this with a code. If the PGA has a notification it returns a value >0, if there is no notification the value remains 0.

Press IN /OUT mode
This Bool is used to indicate which press mode the PGA is in. If the pressInMode bool is set to true then the unit is ready to press in a collet and the pressOutMode bool is set to false. If the bool signals are the other way round, the PGA is ready to eject a collet.

Ready for Commands
This Bool is used to check if the PGA is ready for Commands. If the PGA is ready for Commands the PGA will respond with True. If the PGA is not ready for Commands the PGA will respond with False.

Command received

This Bool is used to check if the PGA has recieved a Command. If the PGA has recieved a Command the PGA will respond with True. If the PGA has not recieved a Command the PGA will respond with False.

At working
This Bool is used to check if the PGA is at working. If the PGA is at working the PGA will respond with True. If the PGA is not at working the PGA will respond with False.

PGA is closed
This Bool is used to check if the PGA is closed. If the door is closed and locked with the bolt the PGA will respond with True. If the PGA is not closed the PGA will respond with False.

PGA is open
This Bool is used to check if the PGA is open. If the door is unlocked and open the PGA will respond with True. If the door is not open the PGA will respond with False.

PG in APG-Unit
This Bool is used to check if the PGA has a Toolholder in the APG. If the PGA has a Toolholder in the APG, the PGA will respond with True. If the PGA has no Toolholder in the APG the PGA will respond with False.

Paused
This Bool is used to check if the PGA has been paused. If the PGA is paused, it responds True. If the PGA is not paused, it responds with False.

Command rejected
This Bool is used to check if the PGA rejected a command. If the PGA will respond with True a command was rejected. The True state is one second active and goes back to false. If the PGA will respond with False everything is ok.

Command finished
This Bool is used to check if the PGA has finished a Command. If the PGA has Command Finished, the PGA will respond with True, the state stays für one second active and goes back to false. If the PGA respond with False the process is not finished yet.

Press finish
This Bool is used to check if the PGA has finished the press. If the pressing is finished, the PGA responds with True, the state remains active for one second and then changes back to False. The process is not finished if the PGA returns False.

PGA is Init
This bool is used to check if the PGA is initialized. If the PGA is initialized, in Press-In mode and the door is open, the PGA will respond with True. If the PGA is in NoInit mode and has no open PGA signal, it responds with false.

- **Write**

Enable Device
This Bool is used to enable the PGA. If the PGA is enabled the PGA will respond with True. If the PGA is not enabled the PGA will respond with False.

Space Free
This Boolean is used to check whether the PGA door and bolt can be moved. If this Boolean is not activated, the door will not move, the PGA will press to the end and pauses all commands.

Execute (Command)

This Bool is used to execute a command. If the PGA has executed a command, the PGA responds with the variable “command received” with True. If the PGA has not executed a command, the PGA responds with the variable “command received” with False.

Reset
This Bool is used to reset the PGA by toggling the variable to true for more than >1s. All states and errors/warnings are reset.

Pause
This Bool is used to pause the PGA. If it is paused, the PGA will respond with the variable "paused" with True and will press to the end and pause all commands. If the PGA is not paused, the response to the variable “paused” will be false.

Clear
If the device has had a fence open error, the PGA must be reactivated by toggling the variable to true for more than >1s, or via the HMI interface. In addition, this bool resets all messages except errors but does not terminate any process and does not reset any states.

Shutdown
Press In Complete
This Boolean is used to press in a collet, close the door by itself if it isn't already closed, press in the collet and open the door all in one hole cycle. This Boolean is only used for the activation of the command "Press in Complete". To start the Press in Complete cycle, the Execute Boolean must be set to True. If the PGA has fully pressed in the collet, the PGA will respond to the PressFinish Bool with True. If the PGA has not fully pressed the collet, the PGA responds with False. If the PGA has finished the hole cycle, the PGA responds to the commandFinished Bool with True. If the PGA has not pressed the collet and has not opened the door, it responds with False.

Press Out Complete
This Boolean is used to press out a collet, close the door by itself if it isn't already closed, press out the collet and open the door all in one hole cycle. This Boolean is only used for the activation of the command "Press out Complete". To start the Press out Complete cycle, the Execute Boolean must be set to True. If the PGA has fully pressed out the collet, the PGA will respond to the PressFinish Bool with True. If the PGA has not fully pressed out the collet, the PGA responds with False. If the PGA has finished the hole cycle, the PGA responds to the commandFinished Bool with True. If the PGA has not pressed out the collet and has not opened the door, it responds with False.

Press In Incomplete
This Boolean is used to press in a collet, close the door by itself if it isn't already closed, press in the collet all in one hole cycle, but not open the door after pressing in. This Boolean is only used for the activation of the command "Press in Incomplete". The Execute Boolean must be set to True to start the Press in Incomplete cycle. If the PGA has fully pressed in the collet, the PGA will respond to the PressFinish Boolean with True. If the PGA has not fully pressed the collet, the PGA will respond with False. If the PGA has finished the hole cycle, the PGA responds to the commandFinished Bool with True. If the PGA has not pressed the collet, it responds with False.

Press Out Incomplete
This Boolean is used to press out a collet, close the door by itself if it isn't already closed, press out the collet all in one hole cycle, but not open the door after pressing out. This Boolean is only used for the activation of the command "Press out Incomplete". The Execute Boolean must be set to True to start the Press out Incomplete cycle. If the PGA has fully pressed out the collet, the PGA will respond to the PressFinish Boolean with True. If the PGA has not fully pressed out the collet, the PGA will respond with False. If the PGA has finished the hole cycle, the PGA responds to the commandFinished Bool with True. If the PGA has not pressed out the collet, it responds with False.

Close PGA

With this Bool, the PGA door can be closed and will lock itself with the bolt. This Boolean is only used to activate the Close PGA command. To start the Close PGA cycle, the Execute Boolean must be set to True. When the PGA has closed the door and locked it with the bolt, the PGA will respond to the PGAisClosed Bool with True. If the PGA has finished closing and locking the door, the PGA responds to theFinished Bool command with True. If the PGA has not closed and locked the door, the PGA responds with False.

Open PGA
With this Bool, the PGA door can be opened and unlock itself. This Boolean is only used to activate the Open PGA command. To start the Open PGA cycle, the Execute Boolean must be set to True. When the PGA has opened the door and unlocked the bolt, the PGA will respond to the PGAisOpen Bool with True. If the PGA has finished open and unlocking the door, the PGA responds to theFinished Bool command with True. If the PGA has not opened and unlocked the door, the PGA responds with False.

Mode Change
This Bool can be used to change the mode of the PGA to the opposite mode. This Boolean is only used for the activation of the "Change Mode" command. To start the Mode Change cycle, the Execute Bool must be set to True. If the PGA has changed the Mode, the PGA will respond to the opposite press mode (pressInMode/pressOutMode) Bool with True. If the PGA has finished changing the Mode, the PGA responds to the commandFinished Bool with True. If the PGA has not changed the Mode, the PGA responds with False.

Mode Change with Door
This Boolean can be used to change the mode of the PGA to the opposite mode and close the door by itself if it isn't already closed, change the mode and open the door all in one hole cycle. This Boolean is only used to activate the "Change Mode with Door" command. To start the Mode Change cycle, the Execute Boolean must be set to True. If the PGA has changed the mode, the PGA will respond to the opposite press mode (pressInMode/pressOutMode) Bool with True. If the PGA has finished the mode change and opened the door, the PGA responds to the commandFinished Bool with True. If the PGA has not changed the mode and has not opened the door, the PGA responds with False.

Mode Change to In
This Bool can be used to change the mode of the PGA to In mode if it isn't already in In mode. This Boolean is only used to activate the "Mode Change to In" command. To start the Mode Change to In cycle the Execute Bool must be set to True. If the PGA has changed mode to In or was already in In Mode, the PGA will respond to the pressInMode Bool with True. If the PGA has finished the mode change or was already in the In Mode, the PGA responds to the commandFinished Bool with True. If the PGA is not in the Press In Mode, the PGA responds with False.

Mode Change to Out
This Bool can be used to change the mode of the PGA to Out mode if it isn't already in Out mode. This Boolean is only used to activate the "Mode Change to Out" command. To start the Mode Change to Out cycle the Execute Bool must be set to True. If the PGA has changed mode to Out or was already in Out mode, the PGA will respond to the pressOutMode Bool with True. If the PGA has finished the mode change or was already in the Out mode, the PGA responds to the commandFinished Bool with True. If the PGA is not in the Press Out Mode, the PGA responds with False.

5. Procedures

In this chapter, we will explore the procedures of the PGA 9500 device over a network interface. The PGA 9500 offers various features and functions that can be accessed and controlled via an API. This section provides a comprehensive overview of the procedures involved in enabling and disabling the PGA over the network, retrieving its status, sending commands, handling errors, changing operating modes, managing collets and holders with a robot and controlling the PGA in detail. Each procedure is accompanied by diagrams and explanations to facilitate a clear

understanding of the API functionality. By following this documentation, users will be able to interact effectively with the PGA 9500 and use its features for their intended applications.

Continuing from the previous introduction, let's explore how the PGA 9500 device interacts with the API through read and write Boolean variables. When working with the PGA 9500 device, users can access various read Boolean variables to retrieve specific status information. These variables act as indicators that provide data related to the device's state. For example, users can query the operational mode read Boolean variable to determine the current mode in which the PGA 9500 is operating. Other read variables may include configuration settings, sensor readings, or diagnostic information. By accessing these read Boolean variables and retrieving their values, users can gain valuable insights into the device's status and make informed decisions based on the received information. To initiate actions or modify settings on the PGA 9500 device, users employ write Boolean variables. Before executing a command, users need to select the corresponding command Boolean variable that represents the desired action. This selection specifies the operation to be performed on the device. Once the command Boolean is set, users trigger the execute Boolean variable, which initiates the execution of the command. This two-step process ensures that commands are intentionally initiated and helps prevent accidental actions. During the execution of a process, the PGA 9500 device provides feedback on the status of the operation. This feedback is typically reflected in other Boolean variables, allowing users to monitor the progress of their actions. For example, a start Boolean variable may indicate that a process has begun, while a success Boolean variable may confirm that the process was completed successfully. Similarly, if a process fails, an error Boolean variable may be set to indicate the failure. By monitoring these status variables, users can track the progress of their operations and respond accordingly. By understanding the interaction between read and write Boolean variables, users can effectively communicate with the PGA 9500 device through the API. This comprehensive documentation provides detailed explanations and examples to guide users in leveraging these features. By utilizing the read Boolean variables, users can retrieve valuable status information from the device. By setting the appropriate write Boolean variables and executing commands, users can control and configure the PGA 9500 device to meet their specific requirements. The feedback provided through other Boolean variables ensures users are aware of the status of their operations, facilitating a smooth and efficient interaction with the PGA 9500 unit.
Additionally, it is important to note that only one command can be active at a time when interacting with the PGA 9500 device through the API. This means that before initiating a new command, the previous command must be completed or canceled. By enforcing this restriction, the PGA 9500 ensures that commands are executed in a controlled manner, preventing conflicts or overlapping actions. This design ensures the integrity of the device's operations and avoids any potential issues that may arise from concurrent command execution. To manage the execution of commands effectively, users should carefully track the status of the ongoing commands and ensure that they are completed or canceled before initiating new ones. This can be achieved by monitoring the relevant Boolean variables associated with the command's execution status. The documentation provided in this chapter includes clear explanations and illustrative diagrams for each command, enabling users to understand the sequence of actions required and the associated Boolean variables. By adhering to the principle of having only one active command at a time, users can effectively control the PGA 9500 device and achieve the desired results without encountering conflicts or inconsistencies. By following the guidelines and best practices outlined in the documentation, users can leverage the full potential of the PGA 9500 device and ensure a smooth and reliable integration within their applications.

5.1. Enable PGA over network (Initialization after Reset or Reboot)

This article highlights the significance of enabling PGA over the network and explores the subsequent command execution process, shedding light on the pivotal role it plays in leveraging the device's functionality. Once the device is ready, users can confidently send commands to control and configure the PGA 9500 according to their specific requirements.

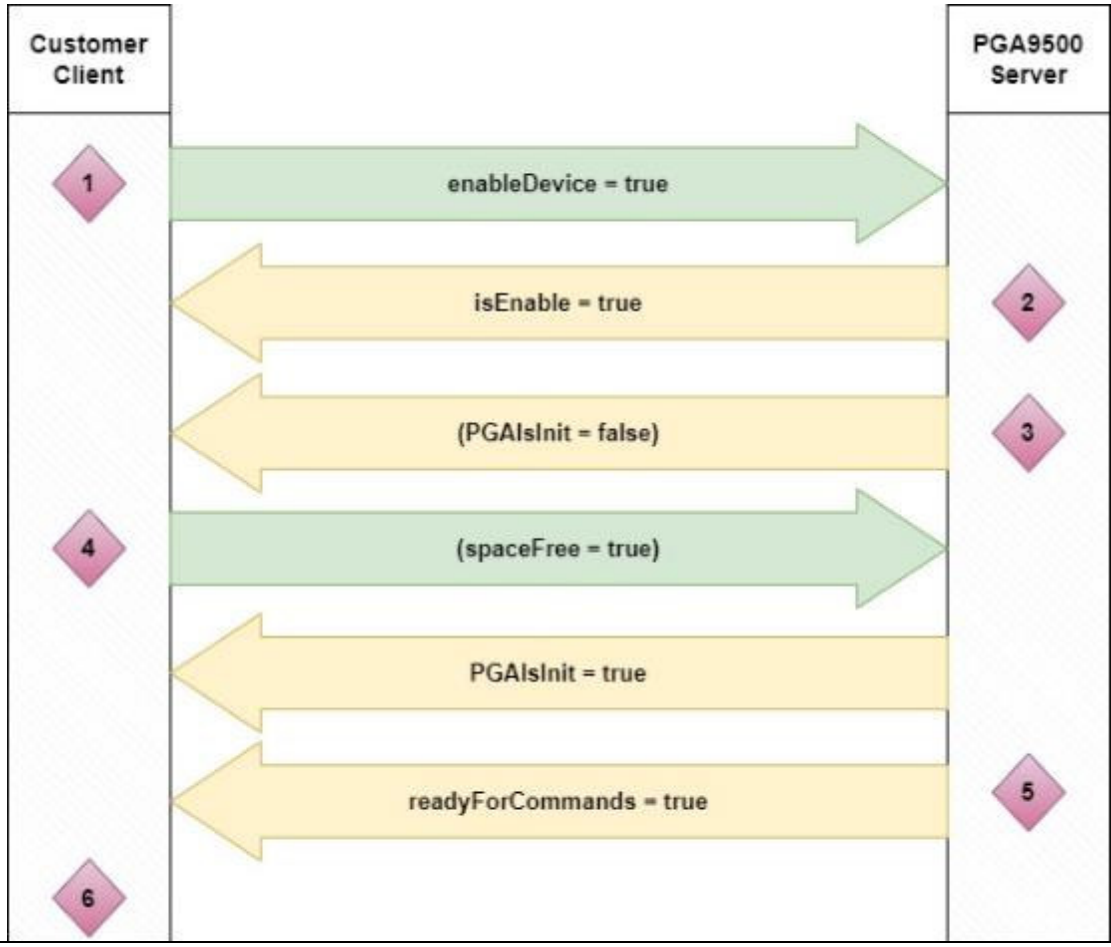

Fig.12: Flowchart of the enabling process (Initialization)

1. The device is powered on and ready to be enabled. The OPC UA connection is established and the device is ready to receive commands. The device is in the disabled state. The device is not ready to execute commands. The enableDevice Boolean variable is set to false and will be set to true to enable the device. If the PGA is not initialized, it is also important, that there is no toolholder placed in the PGA and the PGinAPGunit variable is false. (Initialization is impossible with a toolholder inside the PGA)
2. After setting the enableDevice Boolean variable to true, the device will start the enabling process and will set the isEnabled Boolean variable to true. This indicates that the device is enabled.
3. After rebooting or resetting the device, the device needs to do an initialization before it is ready to receive commands. If the device is already initialized, steps 3 and 4 are not necessary.
4. For the PGA to automatically initialize, the robot or supplier must signal with the variable spaceFree that the door space is free. This is because the door/bolt drives and their sensors also adjust and move during the initialisation process.
5. The PGA is enabled and initialized and sets the readyForCommands variable to true.
6. When this is done by the first time, the device is initialized in the "press In" mode and the "press In" mode is set to true. The device is now ready to receive commands and execute them.

5.2. Disable PGA over network

This article focuses on the process of disabling the PGA9500 device over a network interface, ensuring the safe disconnection and termination of device functionalities. By following a systematic approach, users can initiate the disabling procedure, effectively halting command execution and severing the connection between the PGA9500 and external systems.

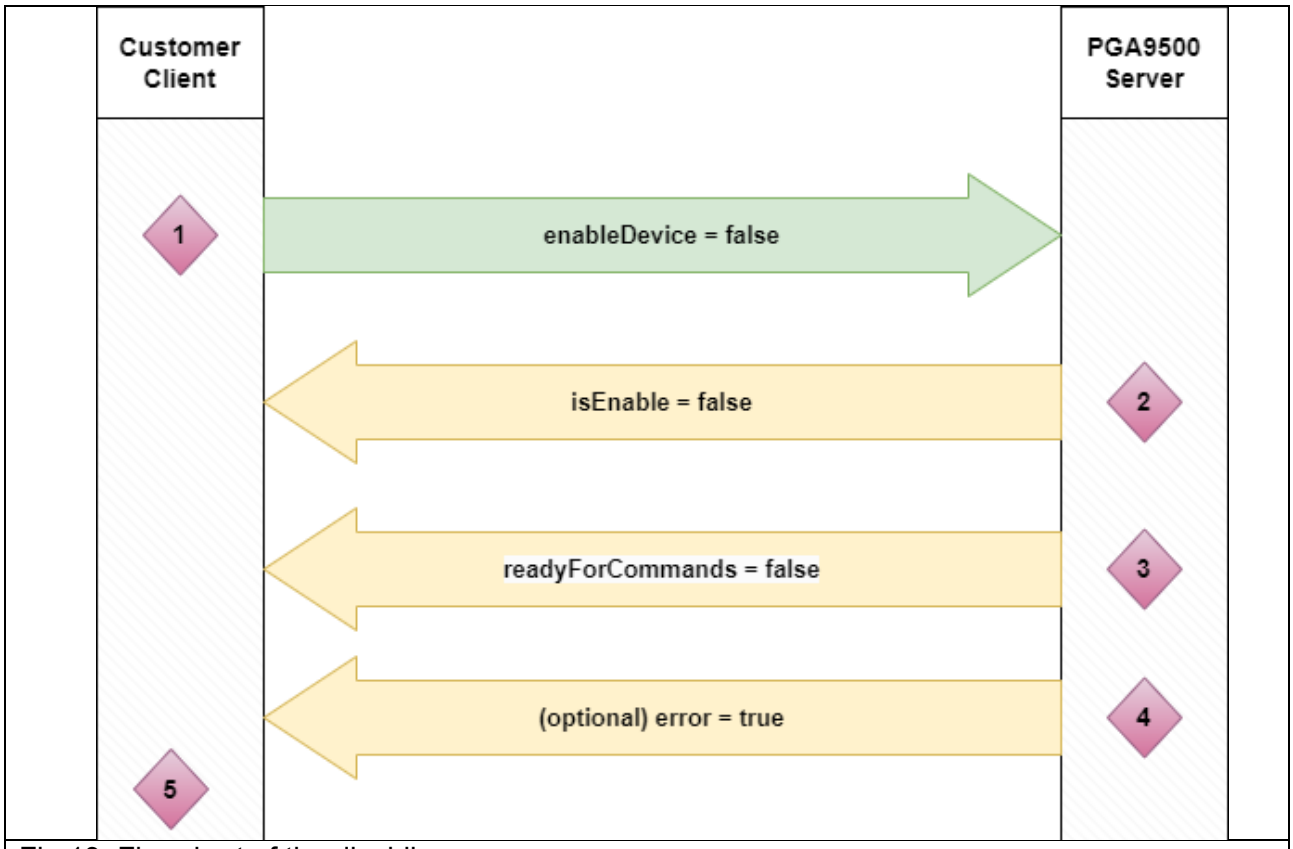

Fig.13: Flowchart of the disabling process

1. The device is enabled and ready to be disabled. The OPC UA connection is established and the device is ready to receive commands. The device is in the enabled state. The device is ready to execute commands. The enableDevice Boolean variable is set to true and will be set to false to disable the device.
2. After setting the enableDevice Boolean variable to false, the device will start the disabling process and will set the isEnabled Boolean variable to false. This indicates that the device is disabled and the initialization process is finished.
3. In addition to the isEnable variable the device will also set the readyForCommands Boolean variable to false. This indicates that the device is not ready to receive commands. The device does not receive commands and execute them.
4. If something goes wrong during the "disable" process, the device will also set the error Boolean variable to true. This indicates that the device is in an error state and the process is not complete. This may also occur if the device was disabled while a process was running. The exact error message can be found in the error table.
5. The Device is now in the disabled and not ready state.

5.3. Get state of PGA over network

This part provides a comprehensive guide on how to retrieve the state information of the PGA9500 device over a network interface. It highlights the various status indicators that can be read from the device, including maintenance status, safety information, error conditions, press-in and press-out

mode, door state, bolt state, and Toolholder presence within the device. By following the instructions outlined in this guide, users can effectively access and interpret these status parameters, enabling them to monitor the device's condition, ensure safety compliance, and make informed decisions regarding maintenance, process control, and Toolholder handling. Understanding the state of the PGA9500 device empowers users to optimize its operation and maximize efficiency in their specific applications.

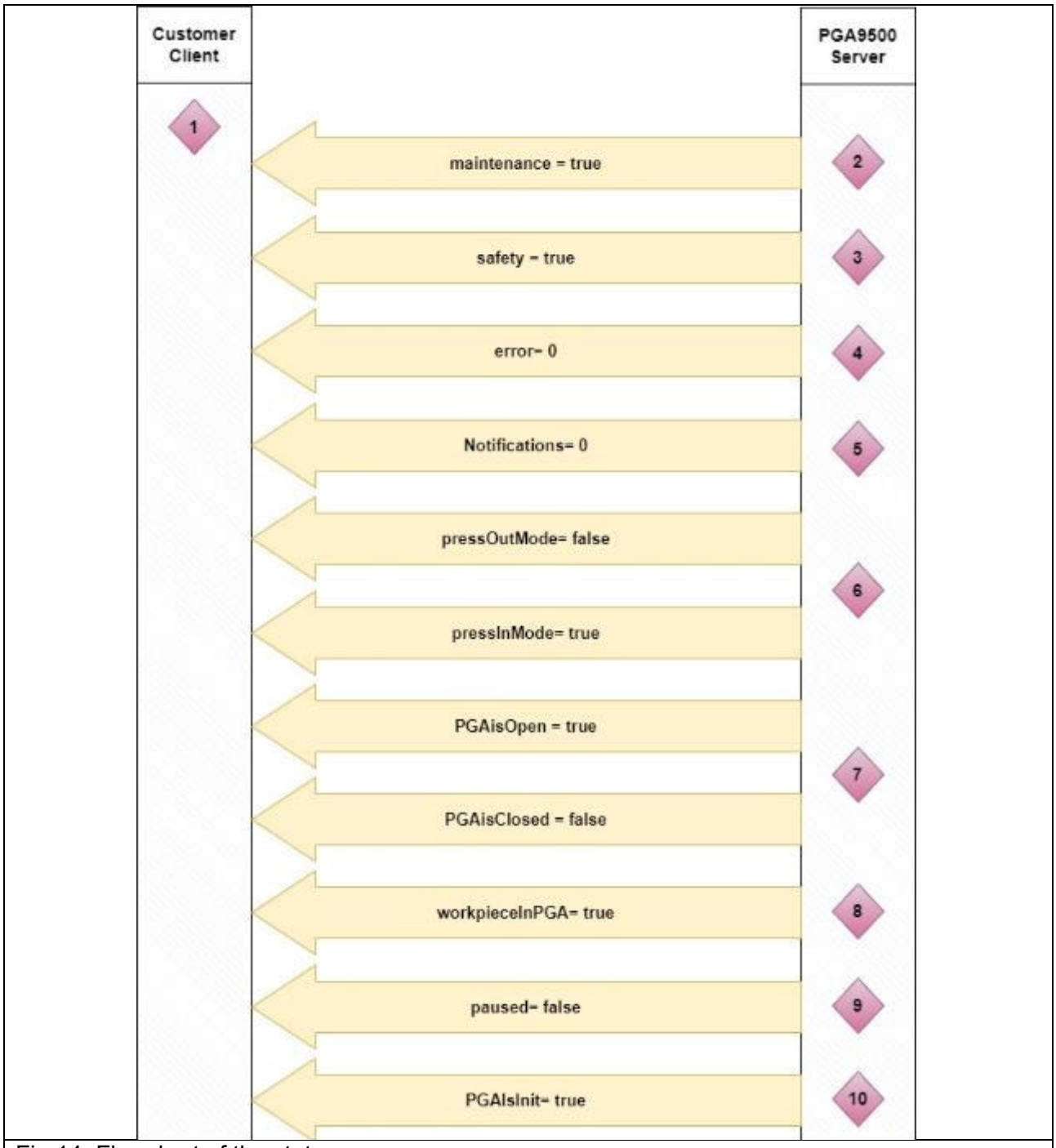

Fig.14: Flowchart of the states

1. All states can be read in any state or at any time when the unit is online, powered on and the OPC UA connection is established. The device must not be enabled to read the state. The device is in the disabled state. The device is not ready to execute commands. The Boolean variable enableDevice is set to false and will be set to true to enable the device.
2. The maintenance boolean indicates, with a positive true, that the device has a maintenance problem that should be rectified. This could be the oil or a seal that needs to be changed.

The exact maintenance message and its description for the associated code can be found in the maintenance table.

3. The safety boolean is true when the safety circuit is closed. This means that the external door circuit is closed and no EMERGENCY STOP button is pressed. This circuit can be open without an error message and if it was open, it must be cleared with the clear boolean to continue.
4. The unsigned integer error is >0 if the unit is in an error state and indicates a specific error with the associated error code. The exact error and its description for the associated code can be found in the error table.
5. The unsigned integer Notifications is >0 if the unit has a notification with the associated notification code. The exact notification and its description for the associated code can be found in the notification table.
6. The pressOutMode boolean indicates the mode of the device. If the boolean is true, the device is in press out mode. If the boolean is false, the device is in the press in mode. If the pressInMode boolean is true, the device is in the press in mode and the pressOutMode boolean is false.
7. The PGAisClosed boolean indicates the status of the door and bolt. If the boolean is true, the door is closed and locked with the bolt. If the boolean is false, the bolt and door are open and the boolean PGAisOpen is true.
8. A special indicator is the workpiceInPGA boolean. This indicates when a tool is inside the PGA. This can be useful to avoid collisions after a reset or initialization. If the variable is true, a tool is in the PGA. If the variable is false, there is no tool in the PGA and the space is free.
9. The pause boolean indicates whether the device has been paused. If the boolean is true, the device is paused and, if started, completes the pressing, then stops and pauses all commands. If the boolean is false, the device operates normally and is not paused.
10. The PGAisInit boolean indicates whether the PGA is being initialized. If the boolean is true, the device is initialized. If the boolean is false, it is not initialized and not ready for commands. The first time it is enabled with free space, it will initialize itself.

5.4. Commands to PGA over network

This comprehensive guide outlines the process of communicating with the PGA 9500 unit over a network interface using variables. It provides a detailed explanation of the step-by-step sequence required to send commands effectively, covering optional elements in the process. The guide encompasses 12 essential steps, including selecting a command boolean, executing the command, and receiving confirmation. By following these instructions, users can navigate the communication process seamlessly, ensuring accurate command transmission and facilitating efficient control of the PGA 9500 unit. Understanding the command structure and adhering to the prescribed sequence empowers users to interact with the PGA 9500 effectively, unlocking its capabilities for a wide range of applications.

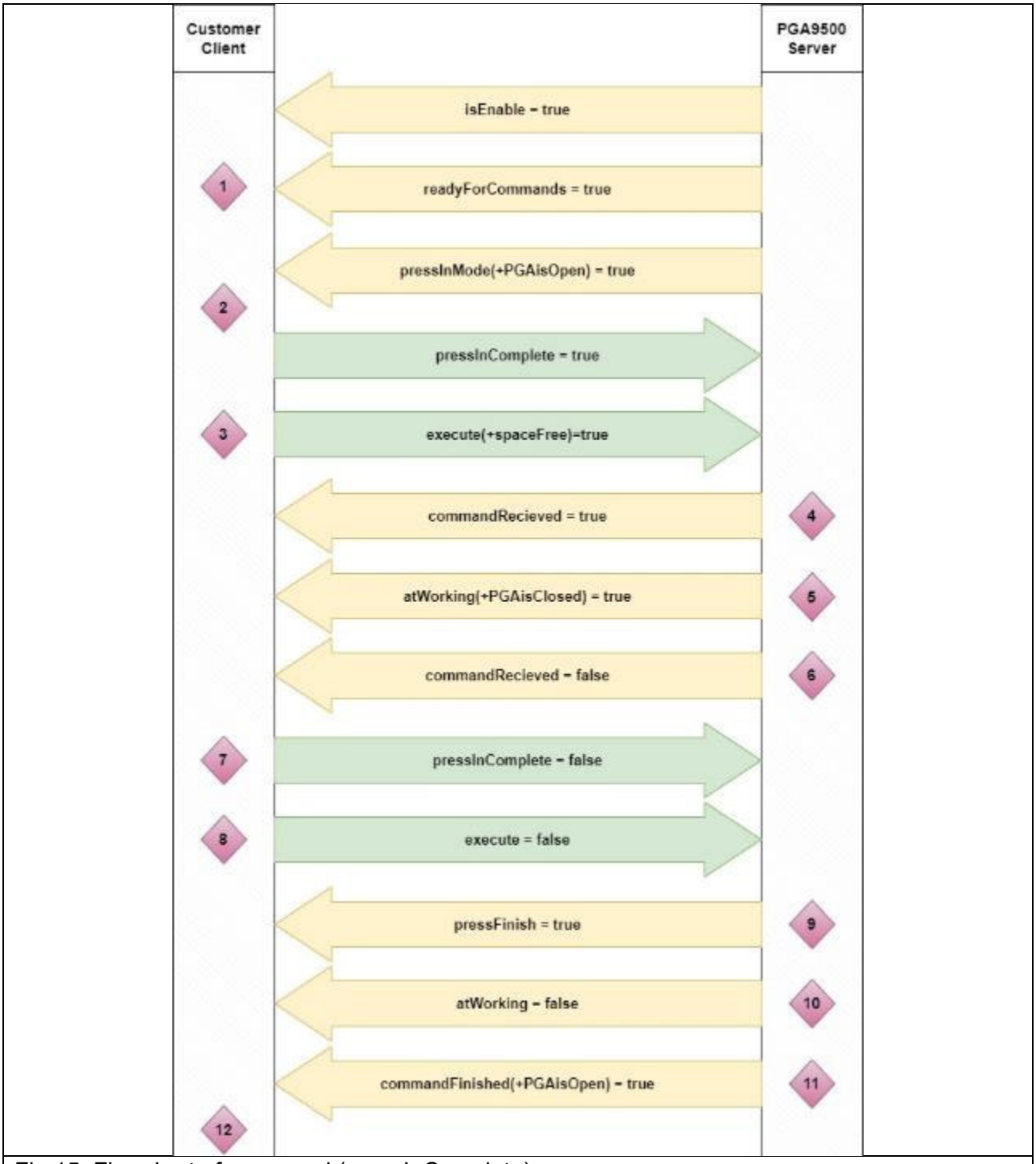

Fig.15: Flowchart of command (pressInComplete)

1. Before a command can be executed or set, the device have to be enabled like described in the enable section. The opc ua connection is established and the device is ready to receive commands. Important is that the isWorking variable is on false and the readyForCommands variable is on true.

2. When the preconditions are fulfill, any command can be set. The commands can be like in the example above pressInComplete to true. There could only be set one command at the same time. When there are more than one command set, the device cannot manage them and rise an error. So it is important to set only one command at the same time. The list of the commands are in the table above in the command section.

3. For the example pressInComplete the PGA has to be in the pressInMode (details to set the right mode in the topic Mode Change). If one command is selected, the execute variable can be set to true. This will execute the command. If the PGAisOpen variable is true, its necessary to set the spaceFree variable to true, so the PGA is allowed to close the door. If its already closed, the spaceFree variable is not relevant.
4. After that command is executed, the PGA register that command and when everything is fine to execute, the commandRecieved value is set to true. That indicates that the PGA accept the command and will execute it. Additionally, the variable readyForCommands is set to false. That indicates that the PGA is not ready to receive another command.
5. With the atWorking variable the PGA indicates that the PGA is in process and works at a command that is set. In the example above, the door of the device will close and the PGAisClosed variable will be set to true, if it wasn't already.
6. After 1-2 seconds the commandRecieved variable is set to true, the PGA turns the commandRecieved variable to false. That is for only for reseting the state of the commandRecieved variable.
7. While the command is executing and the commandRecieved variable has been confirms that the command is working, it is possible to set the command to false.
8. The same applies for the execute variable.
9. When the PG-Unit is finished with the pressing of the collet itself, the PGA sets the pressFinish variable to true.
10. When the PGA is finished with the hole command, the isWorking variable is set to false and indicates that the PGA is no more working at a command.
11. In the example the PGAisOpen will be set to true after the door has opened automatically (not in press*Incomplete). An additional indicator is the commandFinished variable. When this variable turns to true the command is safely and correct finished. The process is done and has no errors or issues. This variable changes back to false after a few seconds.
12. When everything works fine, the readyForCommands variable is set to true and the PGA is ready to receive commands again. When some errors appear while executing a command the PGA indicates this with the error variable as a code and the details are descripted in the Error-Table below.

5.5. Mode change PGA over network

This article explores the mode change functionality of the PGA9500 device over a network interface. The focus is on the two available modes: Press In Mode and Press Out Mode. Users can switch between these modes by employing a similar command structure as described in the previous chapters. Three different methods for mode switching are discussed:

1. transitioning directly to the other mode,
2. transitioning to a specific mode (e.g., Press In Mode)
3. transitioning directly to the other mode while also closing the door.

By following the instructions provided, users can seamlessly change modes and adapt the PGA 9500 unit to their specific operational requirements. Understanding the mode change process enables efficient utilization of the PGA 9500 unit capabilities, enhancing overall productivity and flexibility. This article serves as a comprehensive guide to successfully navigate the mode change procedure for optimal control and performance of the PGA 9500 unit.

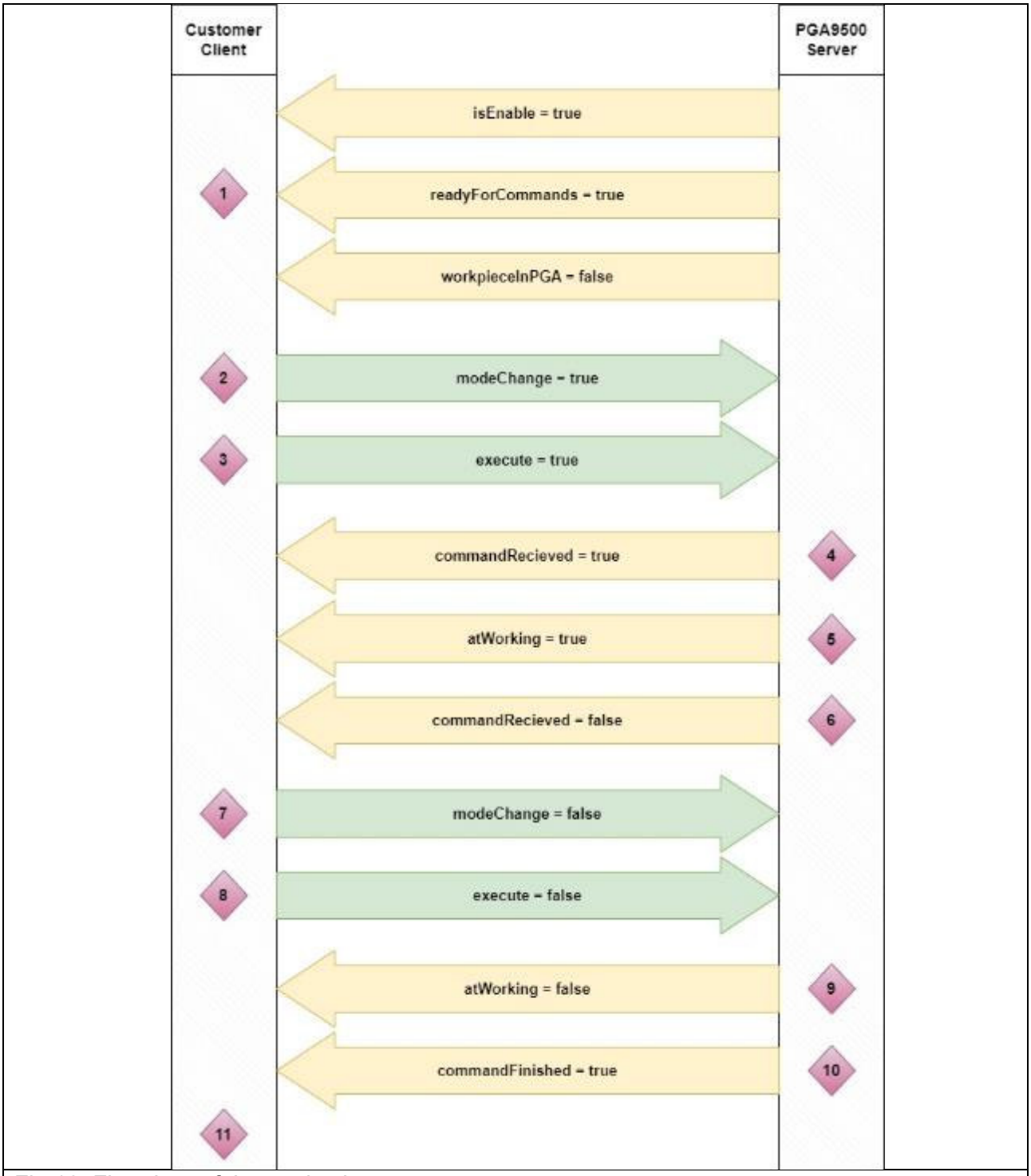

Fig.16: Flowchart of the mode change

1. Before a mode change can be executed or set, the unit has to be enabled like described in the enable section. The OPC UA connection is established and the device is ready to receive commands. Important is that the atWorking variable is on false, the readyForCommands variable is on true and no toolholder is placed in the PGA. It's impossible to change the mode with a toolholder inside.

2. If the preconditions are fulfilled, any mode change can be set. The mode changes can be like in the picture above pressInMode to true. There could only be set one mode change at the same time. When there are more than one mode change set, the device cannot manage them and rise an error. So, it is important to set only one mode change at the same time. The list of the mode changes is in the table above in the mode change section.

3. If one mode change is selected, the execute variable can be set to true. This will execute the mode change.
4. After that mode change is executed, the PGA register that mode change and when everything is fine to execute, the commandRecieved value is set to true. That indicates that the PGA accept the mode change and will execute it. Additionally, the variable readyForCommands is set to false. That indicates that the PGA is not ready to receive commands.
5. With the atWorking variable the PGA indicates that the PGA is in process and works at a mode change that is set.
6. After 1-2 seconds the commandRecieved variable is set to true, the PGA turns the commandRecieved variable to false. That is only for resetting the state of the commandRecieved variable.
7. While the mode change is being executed and the commandReceived variable has confirmed that the mode change is working, it is possible to set the mode change to false.
8. The same applies for the execute variable.
9. If the PGA is finished with the given mode change, the isWorking variable is set to false and indicates that the PGA is no more working at a mode change.
10. An additional indicator is the commandFinished variable. If this variable turns to true the mode change is safely and correct finished. The process is done and has no errors or issues. This variable changes back to false after a few seconds
11. If everything works fine, the readyForCommands variable is set to true and the PGA is ready to receive mode changes again. The mode change has worked successful and the PGA is ready for the next mode change. If some errors appear while executing a mode change the PGA indicates this with the error variable and a specific code.

5.6. Control PGA detailed over network

To initiate the process, an external device, such as a robot or a linear axis system, starts by inserting a PG holder into the PGA9500. The PG holder contains the necessary tool and gripper for the intended operation. Once the holder is securely placed, space of the door is free and the PGA9500 is closed, signaling the device to proceed with the subsequent steps. Upon detecting the closed PGA9500, it allows the robot to release its grip on the toolholder. During this phase, the automation system no longer needs to actively hold the toolholder, as the PGA9500 takes over the responsibility of securing it in place. Before the door reopens, the external automation system must regain control and ensure the toolholder is firmly held and the space of the door is free again. This step is essential to maintain stability and prevent any unwanted movement or dislodging of the tool during the subsequent operation. With the external system holding the toolholder and signaling a free space, the door of the PGA9500 opens, granting access to the inserted tool. The open door allows the robot to safely extract the pressed-in tool from the PGA9500. Care should be taken to ensure proper alignment and secure handling during this step to avoid any damage to the tool or the device. Once the extraction process is completed, the extracted tool is now available for reuse in subsequent operations. The PGA9500 is now ready for the next tool insertion or extraction cycle. It is important to note that this process cycle is applicable to PG holders of all sizes, and it applies to both tool insertion and extraction operations. The synchronized coordination between the PGA9500 device and the external automation system ensures efficient and reliable tool handling throughout the operation.

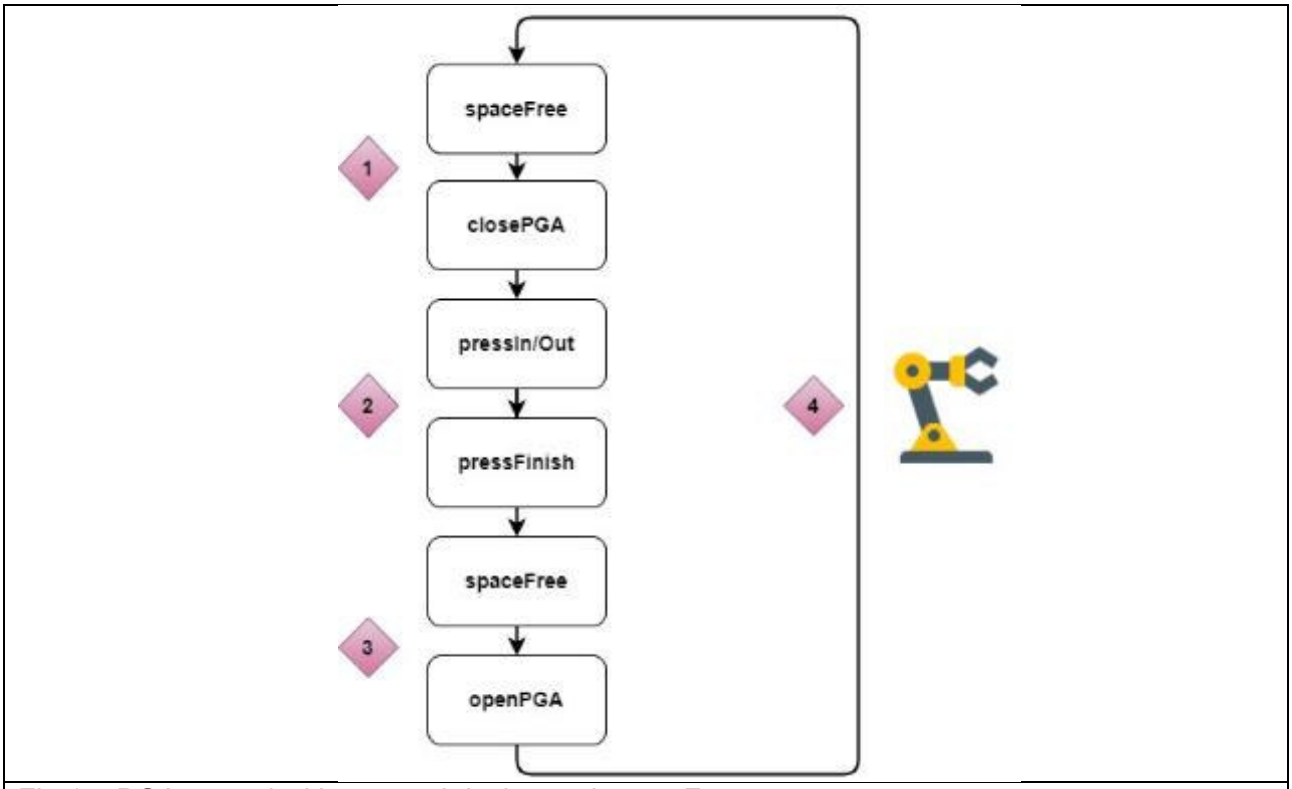

Fig.17: PGA control with external device and spaceFree

1. The PG toolholder is inserted into the PGA9500, the PGA is enabled, initialized and ready to receive any commands. The robot or linear axis system is not in the motion space of the door. The PGA closes automatically or can be closed over network ((closePGA+)pressXComplete, (closePGA+)pressXIncomplete).
2. The PGA9500 can start with the press procedure. It is the same procedure for the press in or for the press out mode. When the press procedure is finished, the PGA9500 indicates that the press procedure is finished.
3. The robot or linear axis system is not in the motion space of the door. The PGA opens automatically or can be opened over network (pressXComplete, pressXIncomplete+openPGA). For the automation system around the PGA9500 it is important to hold the PG toolholder otherwise the PG toolholder could fall out of the PGA9500.
4. When this process is finished, the robot can interact with the other storage- or handling-systems and the PGA9500 is ready for the next press procedure.

5.7. Change toolholder with robot feedback

One of the remarkable features of the PGA9500 device is its ability to provide real-time feedback on the status of the inserted tool. This ensures effective monitoring and enables users to stay informed about the presence or absence of a tool within the PGA9500. To initiate the tool status monitoring process, a robot is utilized to insert a tool into the PGA9500. The device is equipped with a sensor that actively detects the presence of the tool. As the robot places the tool into the PGA9500, the sensor feedback becomes active, indicating that the tool is securely inserted and ready for operation. Once the tool is inserted, the sensor feedback remains active, providing continuous monitoring of the tool's presence within the PGA9500. This real-time status information can be accessed through the API, allowing users to retrieve the tool status as needed. When the robot removes the tool from the PGA9500, the sensor feedback is deactivated, indicating that the tool is no longer present in the device. This change in status can be monitored through the API, providing prompt updates to the external automation system.

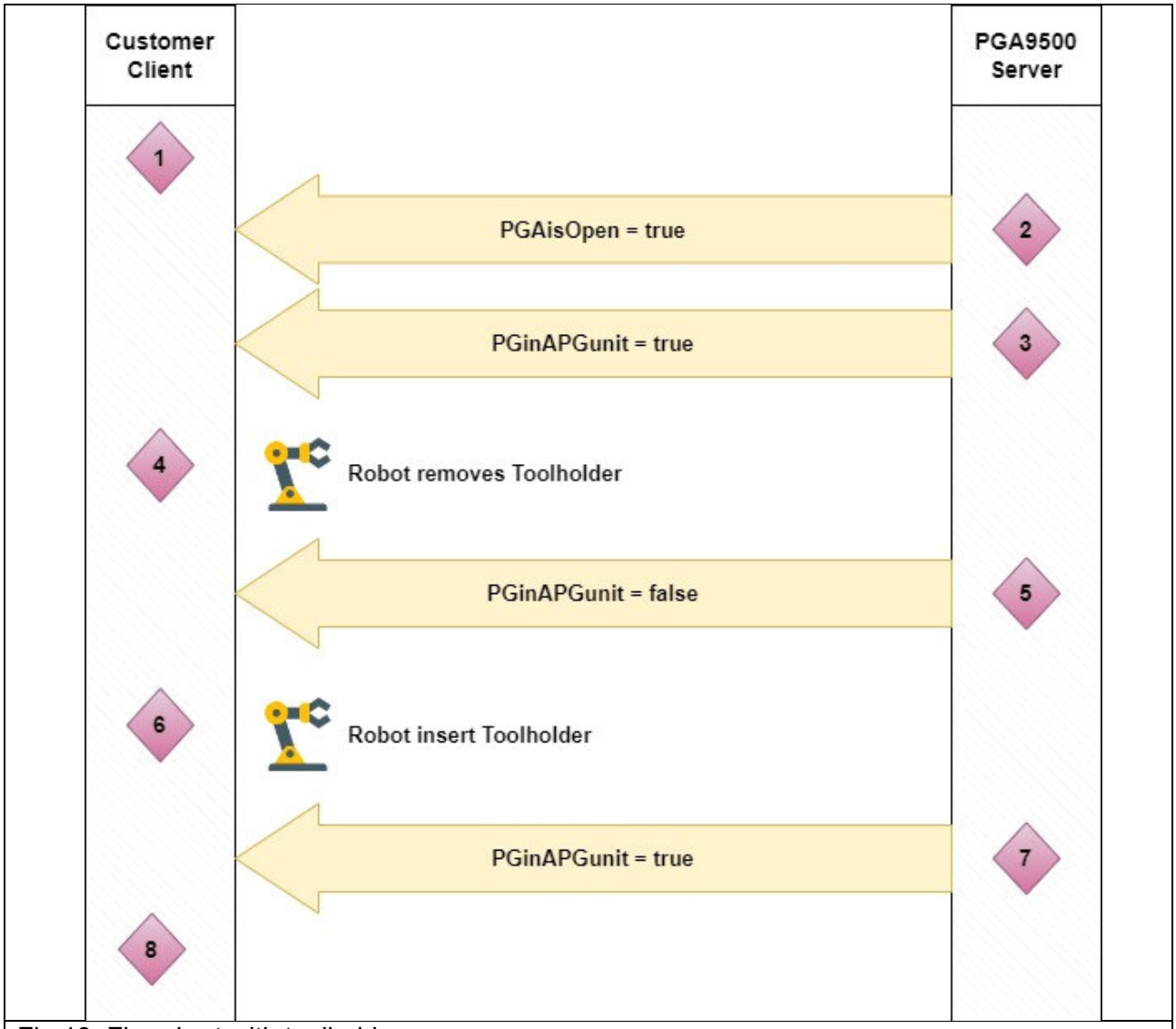

Fig.18: Flowchart with toolholder sensor

1. The device is enabled and ready to receive any commands. In the device is placed a tool-holder and the process of pressing in or out is completed.
2. The PGAisOpen variable is on true, that means the door is open and the robot can interact with the PGA9500. In addition, the spaceFree variable could be set to false, so that it is impossible for the door to move and the robot can safely place a new tool-holder in the PGA9500.
3. On the OPC UA interface is the PGinAPGunit true, because a toolholder is in the PGA9500 and can be gripped by the robot.
4. The robot can grab the toolholder and can move with a linear line interpolation out of the PGA9500.
5. When the toolholder is removed from the PGA9500 the PGinAPGunit variable turn to false and signaling that no toolholder is in the device.
6. The robot can change the tool or the holder and place it into the PGA 9500.
7. As soon as the Toolholder is laid back in the PGA 9500 the variable turns back to true and signaling that the device can interact with the toolholder.

8. The device is ready to receive new commands and start any process with the new tool-holder. Remember, the robot can only let the tool-holder go, when the PGA 9500 is closed.

5.8. Safety handling

Another notable feature of the PGA 9500 is its ability to handle an interruption in the safety circuit. The safety concept ensures a safe stop of all relevant systems without having to completely leave the process. This ensures effective monitoring and allows the user to resume operation from the same point in the event of an interruption during operation. The PGA 9500 provides an externally switchable safety circuit that is continuously monitored. The user can access the status of the safety circuit via the API. The user is also informed of further steps via notification signals. During a press process, the PGA 9500 monitors whether the process has been completed. If it has been aborted by the safety during the process, the device recognizes that it needs to perform that specific process step again. This ensures that the process can continue after an interruption without any loss of information and that the pressing process is fully completed. All the user has to do is confirm the interruption for the PGA 9500 to continue the process.

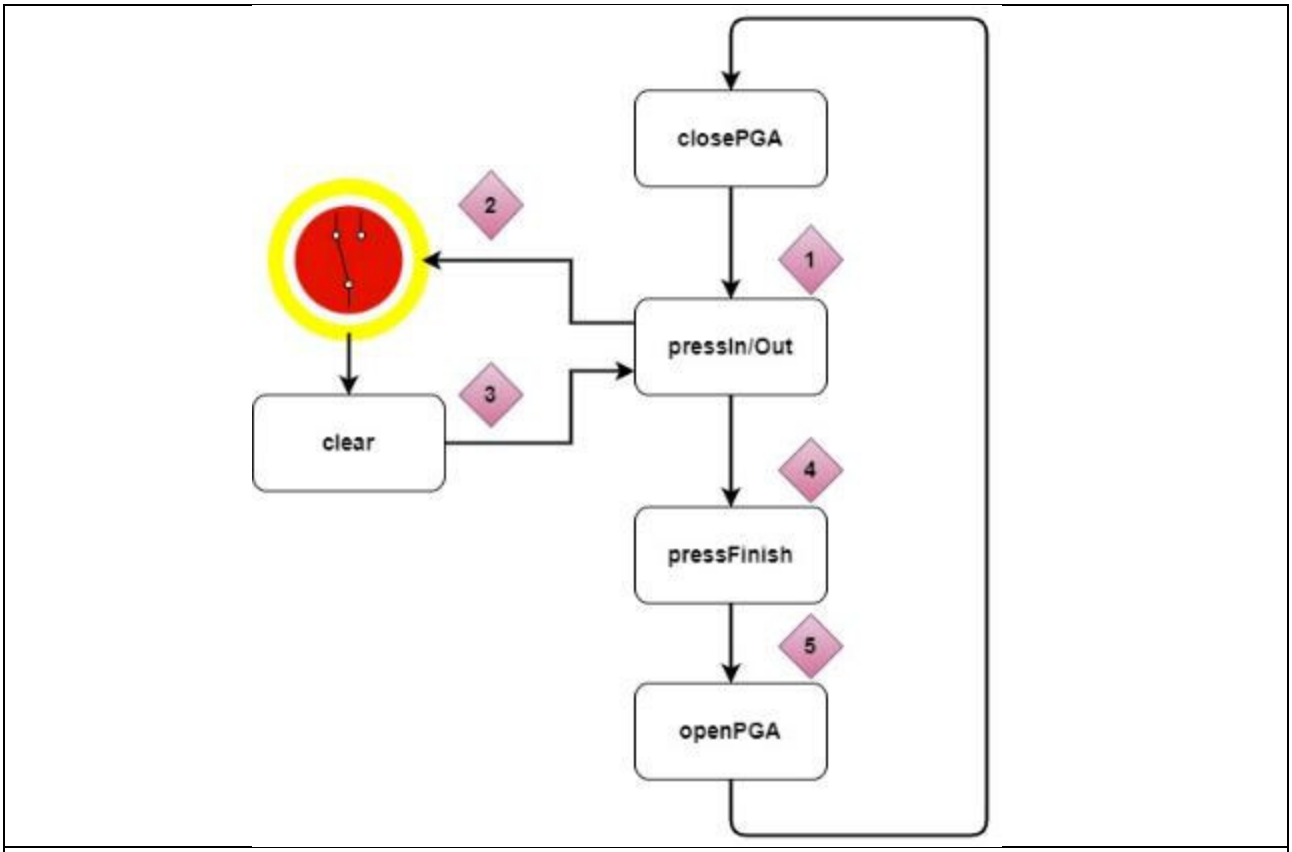

Fig.19: Safety handling flowchart

1. The PG toolholder is inserted into the PGA 9500, the PGA is enabled, initialized and ready to receive any commands. The robot or linear axis system is not in the motion space of the door. The PGA is closed and receives the command to start a pressing process.
2. The PGA 9500 will start the press process. The procedure is the same for press in and press out. This is stopped by interrupting the safety circuit before completion and receiving the true signal from the pressFinish variable. This does detect that the collet is not completely pressed in/out. The safety variable is set to false.
3. After leaving the security area and closing the security circuit again, the safety variable is set to true. To continue the process, this must be confirmed by setting the variable clear to true for >1s. The press cycle has not been completed. The PG press will move to the initialization position and then continue the process.

(If the safety circuit is opened after the PGA has been closed but the pressing process has not yet been started, the door opens and closes again when it is restarted.)

4. The press process is fully completed and the pressFinish variable is set to true.
5. When the safety circuit was triggered, the power supply to the door operator controls is also interrupted. These require a few seconds after closing the safety circuit to be able to receive commands again. If the door is to be opened immediately after such an event, the process is paused and automatically resumed when the door drives are ready again. The notification variable reports this with the corresponding code.

5.9. Example handling when PGA unit is in error over network

This section describes how to handle an error over the network using the PGA interface. An example of an error event is described. With this step-by-step description, it is possible to troubleshoot an error event and, if possible, complete the failed process. It is important to note that when inspecting close to the PGA9500, the safety circuit needs to be open.

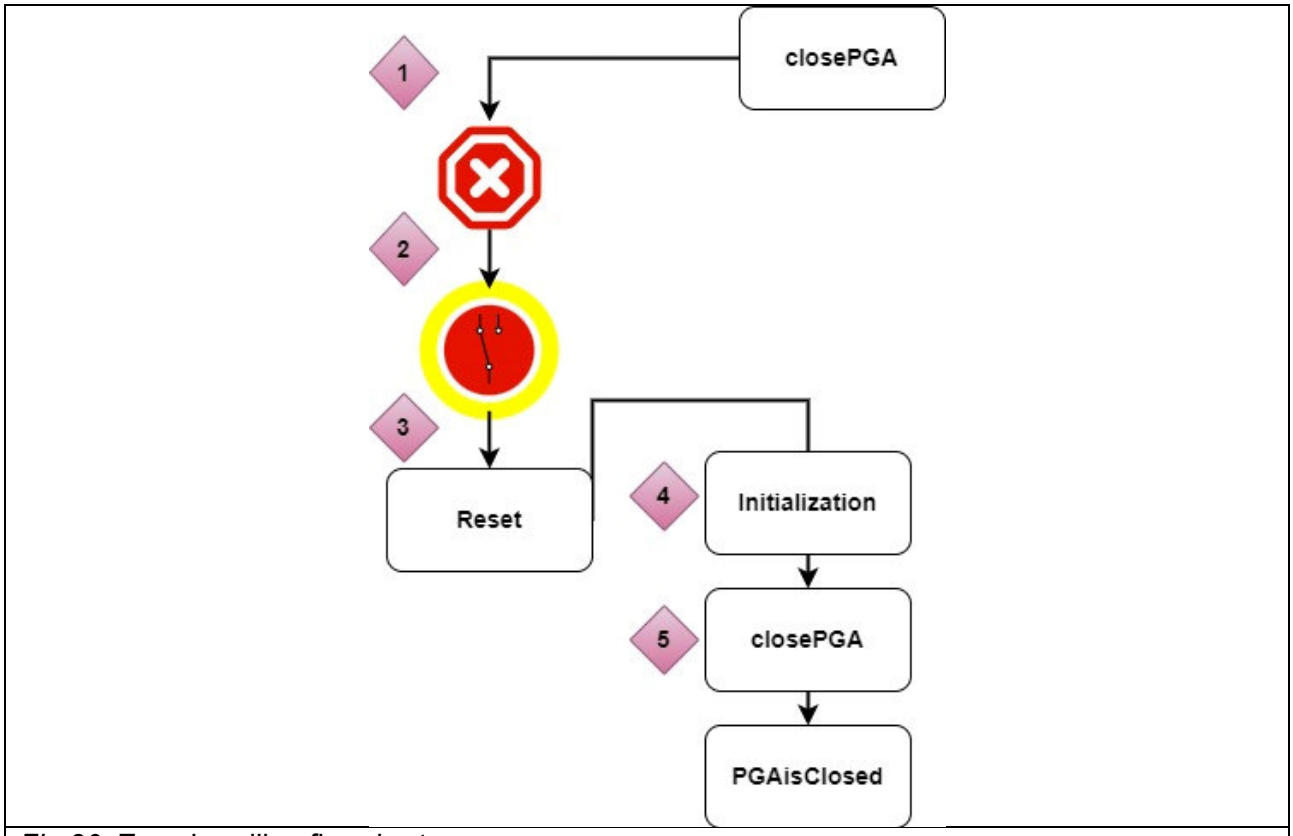

Fig.20: Error handling flowchart

1. The PGA is enabled, initialized and ready to receive any commands. The PGA receives the command to close the door. An error has occurred. The door cannot be closed. The exact error is indicated by a code on the error variable. The exact description of the error code can be found in the error table.
2. Disconnect the safety circuit after a quick check of the PGA9500. If the door was blocked, the drive is now disengaged and the force on the door is turned off. The door can now be moved and the blockage removed.
3. After leaving the security area and closing the security circuit again, the safety variable is set to true. The Reset variable has to be set to true for >1s. It is also recommended to reset

all user commands and signals as well. The error code should be reset and if fixed should no longer appear.

4. After reset, the PGA must be initialized. The enableDevice variable is set to true and the door movement space is reported free by setting the spaceFree variable to true, which starts the automatic initialization. Once initialization is complete, the device is ready to receive and execute commands.
5. The PGA is enabled, initialized and ready to receive any commands, the space of the door is free and the command to close the PGA is sent again. The door is closed and locked.

6. Error handling and notifications

6.1. Error codes

Error Code	Name	Description	Solution
0	No Error	No Error	No Error
1	Door cannot be closed	Door Close timeout error	There must be an Error with the Engine or the power supply. There could also be something between Door and PGA.
2	Door cannot be opened	Door Open timeout error	There must be an Error with the Engine or the Power Supply. There could also be something between Door and PGA.
3	An unknown command was used or the command was misspelled	The command was not recognized by the PGA9500.	Check the command and try again.
4	There is an error in the booting phase	The PGA9500 is not ready to receive any commands.	Check the device and try again (reboot the device).
5	The mode cannot be changed	The mode cannot be changed. Tool inside the PGA	Check the device, remove the holder or update the calibration of the Tool sensor
6	The PGA cannot press in	The PGA cannot press in. Issue with signals or the Press-Unit.	Check the device, the connection and maybe the wires to the PGA.
7	The PGA cannot press out	The PGA cannot press out. Issue with signals or the Press-Unit.	Check the device, the connection and maybe the wires to the PGA.
8	Bolt cannot be closed	Bolt Close timeout error	There must be an Error with the Engine or the Power Supply. There could also be something between Bolt and PGA

Error Code	Name	Description	Solution
9	Bolt cannot be opened	Bolt Open timeout error	There must be an Error with the Engine or the Power Supply. There could also be something between Bolt and PGA
10	Tool inside the PGA while Initialization	Impossible to do initialization with a tool inside the PGA	Check the device, remove the holder or update the calibration of the Tool sensor
11	Bolt Sensor open and close at the same time	Open and Close Sensor are true at the same time	There must be an Error with the Sensors or the Settings of the Sensors/Cylinders are wrong.
12	Door Sensor open and close at the same time	Open and Close Sensor are true at the same time	There must be an Error with the Sensors or the Settings of the Sensors/Cylinders are wrong.

6.2. Notification codes

Maintenance Code	Name	Description	Solution
0	No Error	No Error	No Error
1	The Commands from external are too fast	The Device cannot register the commands at this speed	Send commands with more time interval
2	Wait, the door-space is not free	The external supplier has not reported the door space as free	Report space free so the door can operate
3	Wait (Press cycle time too short)	The press cycle interval is too fast	The machine can perform a pressing cycle every 30 seconds
4	PGA is in the wrong mode	The press mode is wrong for the press command	Change the press command or the press mode
5	Wait until the door drives are rebooted	Door drives has to reboot due to safety	Wait a few seconds until the door operators can execute commands again
6	PGA is not ready or not enabled	The Device is not ready or set enabled	Check if the Device is ready and enabled
7	PG-Press is not Init	The PG-Press is not Init after safety or reset	After safety or reset, the PG-Press is not initialized. Enable and clear, it will do it automatically

Mainte-nance Code	Name	Description	Solution
8	VFD is in Safety mode or has an Error. More Infor-mation is displayed on the VFD.	More Information about the VFD you can read from the display on it.	Check the PG-Device, re-start and try again.
9	Press >>clear<< to con-tinue	Process is stopped due to safety	Confirm with clear to con-tinue the process

7. Examples and code snippets

Link to RFCH Github page to example projects

API Documentation:
https://github.com/REGO-FIX/PGA-9500-API/blob/main/API-PGA9500.md
Example Projects:
https://github.com/REGO-FIX/PGA-9500-API/tree/main/Examples
Beckhoff:
https://github.com/REGO-FIX/PGA-9500-API/tree/main/Examples/Beckhoff
Codesys:
https://github.com/REGO-FIX/PGA-9500-API/tree/main/Examples/Codesys

8. Resources and links

8.1. Rego-Fix

https://ch.rego-fix.com

//TODO https://ch.rego-fix.com/de/PGA9500

8.2. powRgrip ®

https://ch.rego-fix.com/en/products/system/powrgrip

8.3. OPC UA

https://opcfoundation.org/
https://opcfoundation.org/members
https://opcfoundation.github.io/UA-.NETStandard/
https://github.com/OPCFoundation/UA-Java-Legacy/blob/master/examples/basic/README.md
https://infosys.beckhoff.com/index.php?content=../content/1031/tcopcuaser-ver/633776523.html&id=
https://de.codesys.com/produkte/codesys-runtime/opc-ua.html
https://www.rockwellautomation.com/docs/en/factorytalk-optix/1-00/contents-ditamap/developing-solutions/object-examples/opc-ua-client-example.html
https://support.industry.siemens.com/cs/document/109762770/s7-anwenderbaustein-f%C3%BCr-den-opc-ua-client-einer-simatic-s7-1500-?lc=de-de
https://product-help.schneider-electric.com/Machine%20Ex-pert/V1.2/de/m262prg/m262prg/OPC_UA_Server_Client_Configuration/OPC_UA_Server_Client_Configuration-1.htm?rhtocid=_0_18_0
https://opcfoundation.org/members/view/341
https://www.fanuc.eu/ch/de/cnc/connectivity/opc-server
https://industrial.omron.eu/en/solutions/product-solutions/nj5-controller-with-opc-ua